(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 August 2002 (22.08.2002)

PCT

(10) International Publication Number
WO 02/065700 A2

(51) International Patent Classification⁷: H04L 12/00

(21) International Application Number: PCT/GB02/00662

(22) International Filing Date: 14 February 2002 (14.02.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

| | | |
|---|---|---|
| 0103678.9 | 14 February 2001 (14.02.2001) | GB |
| 0103687.0 | 14 February 2001 (14.02.2001) | GB |
| 0121790.0 | 10 September 2001 (10.09.2001) | GB |

(71) Applicant *(for all designated States except US)*: **CLEARSPEED TECHNOLOGY LIMITED** [GB/GB]; Wallscourt Farm, Filton Road, Bristol BS34 8RB (GB).

(72) **Inventors; and**
(75) **Inventors/Applicants** *(for US only)*: **SWARBRICK, Ian** [GB/GB]; 83a Hill Road, Clevedon, North Somerset BS21 7PL (GB). **WINSER, Paul** [GB/GB]; 8 Oakleigh House, Bridge Road, Leigh Woods, Bristol BS8 3PB (GB). **RYAN, Stuart** [GB/GB]; 11 Parsons Way, Wooton Basset, Swindon SN4 8BY (GB).

(74) **Agent: VIGARS, Christopher, Ian**; Haseltine Lake & Co., Imperial House, 15-19 Kingsway, London WC2B 6UD (GB).

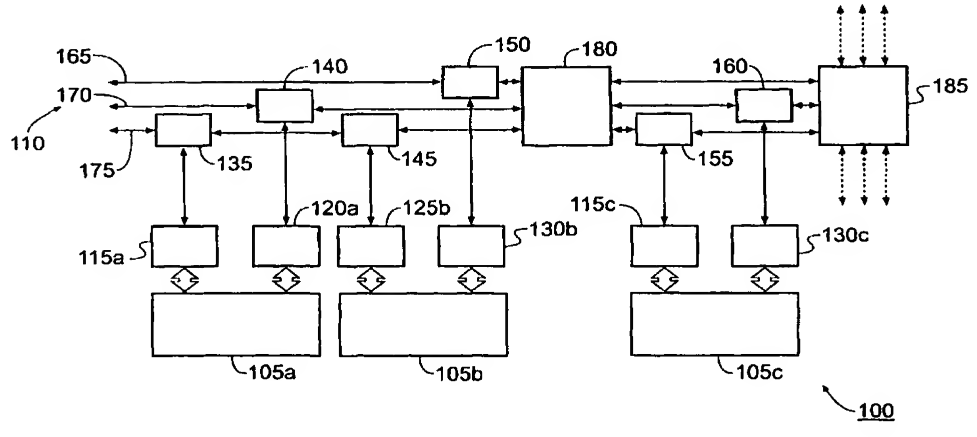
(81) **Designated States** *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

**Published:**
— *without international search report and to be republished upon receipt of that report*

*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

(54) **Title:** AN INTERCONNECTION SYSTEM



(57) **Abstract:** An interconnection system (110) interconnects a plurality of reusable functional units (105a), (105b), (105c). The system (110) comprises a plurality of nodes (135), (140), (145), (150), (155), (160) each node communicating with a functional unit. A plurality of data packets are transported between the functional units. Each data packet has routing information associated therewith to enable a node to direct the data packet via the interconnection system.

WO 02/065700 A2

# AN INTERCONNECTION SYSTEM

## TECHNICAL FIELD

The present invention relates to an interconnection network. In particular, but not exclusively, it relates to an intra chip interconnection network.

## BACKGROUND OF THE INVENTION

A typical bus system for interconnecting a plurality of functional units (or processing units) consists of either a set of wires with tri-state drivers, or two uni-directional data-paths incorporating multiplexers to get data onto the bus. Access to the bus is controlled by an arbitration unit, which accepts requests to use the bus, and grants one functional unit access to the bus at any one time. The arbiter may be pipelined, and the bus itself may be pipelined in order to achieve a higher clock rate. In order to route data along the bus, the system may comprise a plurality of routers which typically comprise a look up table. The data is then compared with the entries within the routing look up table in order to route the data onto the bus to its correct destination.

However, such routing schemes can not be realised on a chip, since the complexity and the size of its components make this infeasible. This has been overcome in existing on-chip bus systems by using a different scheme in which data is broadcasted, that is, transferring the data from one functional units to a plurality of other functional units simultaneously. This avoids the need for routing tables. However, broadcasting data to all functional units on the chip consumes considerable power and is, thus, inefficient. Also it is becoming increasingly difficult to transfer data over relatively long distances in one clock cycle.

Furthermore, in a typical bus system, since every request to use the bus (transactor) must connect to the central arbiter, this limits the scalabilty of the system. As bigger systems are built, the functional units are further from the arbiter, so latency increases and the number of concurrent requests that may be handled by a single arbiter is limited. Therefore, in such central arbiter based bus systems, the length of the bus and the number of transactors are normally fixed at the outset. Therefore, it would not be possible to lengthen the bus at a later stage to meet varying system requirements.

Another form of interconnection is a direct interconnection network. These types of networks typically comprise a plurality of nodes, each of which is a discrete router chip. Each node (router) may connect to a processor and to a plurality of other nodes to form a network topology.

In the past, it has been infeasible to use this network-based approach as a replacement for on-chip buses becauses the individual nodes are too big to be implemnted on a chip.

Many existing buses are created to work with a specific protocol. Many of the customised wires relate to specific features of that protocol. Conversely many protocols are based around a specific bus implementation, for example having specific data fileds to aid the arbiter in some way.

## SUMMARY OF THE INVENTION

The object of the present is to provide an interconnection network as an on-chip bus system. This achieved by routing data on the bus as opposed to broadcasting data. The routing of the data being achieved by a simple addressing scheme in which each transaction has routing information associated therewith, for example a geographical address, which enables the nodes within the interconnection network to route the transaction to its correct destination.

In this way, the routing information contains information on the direction to send the data packet. This routing information is not merely an address of a destination but provides directional information, for example x,y coordinates of a grid to give direction. Thus the nodes do not need routing table(s) or global signals to determine the direction since all the information the node needs is contains in the routing information of the data packets. This enables the circuitry of the node and the interconnection system to be simplified making integrated of the system onto a chip feasible.

If each functional unit is connected to a node, and all nodes are connected together, then a pipeline connection will exist between each pair of nodes in the system. The number of intervening nodes will govern the number of pipeline stages. If there is pair of joined nodes where the distance between them is too great to transmit data within a single clock cycle, a repeater block can be inserted between the nodes. This block registers the data, while maintaining the same protocol as the other bus blocks. The inclusion of the repeater blocks allows interconnection of arbitrary length to be created.

The interconnection system according the present invention can be utilised in an intra-chip interconnection network. Data transfers are all packetized, and the packets may be of any length that is a multiple of the data-path width. The nodes of the bus used to create the interconnection network (nodes and T-switches) all have registers on the data-path(s).

The main advantage of the present invention is that it is inherently re-usable. The implementer need only instantiate enough functional blocks to form an interconnection of the correct length, with the right number of interfaces, and with enough repeater blocks to achieve the desired clock rate.

The interconnection system in accordance with the present invention employs distributed arbitration. The arbitration capability grows as more blocks are added. Therefore, if the bus needs to be lengthened, it is a simple matter of instantiating more nodes and possibly repeaters. Since each module manages its own arbitration within itself, the overall arbitration capability of the interconnect increases. This makes the bus system of the present invention more scalable (in length and overall bandwidth) than other conventional bus systems.

The arbitration adopted by the system of the present invention is truly distributed and 'localised'. This has been simplified such that there is no polling to see if the downstream route is free as in conventional distributed systems, instead this information is initiated by the ‘blocked node’ and pipelined back up the interconnection (bus) by upstream nodes.

The interconnection in accordance with the present invention is efficient in terms of power consumption. Since packets are routed, rather than broadcast, only the wires between the source and destination node are toggled. The remaining bus drivers are clock-gated. Hence the system of the present invention consumes less power.

Furthermore, every node on the bus has a unique address associated with it; an interface address. A field in the packet is reserved to hold a destination interface address. Each node on the bus will interrogate this field of an incoming packet; if it matches its interface address it will route the packet off the interconnection (or bus), if it does not match it will route the packet down the bus. The addressing scheme could be extended to support "wildcards" for broadcast messages; if a subset of the address matches the interface address then the packet is routed off the bus and passed on down the bus, otherwise it is just sent on down the bus.

For packets coming on to the bus, each interface unit interrogates the destination interface address of the packet. This is used to decide which direction a packet arriving on the bus from an attached unit is to be routed. In the case of a linear bus, this could be a simple comparison: if the destination address is greater than the interface address of the source of the data then the packets routed "up" the bus, otherwise the packet is routed "down" the bus. This could be extended to each interface unit such that each node maps destination addresses, or ranges of addresses, to directions on the bus.

Preferably, the interface unit sets a binary lane signal based on the result of this comparison. In this way functionality is split between the node and interface unit. All "preparation" of the data to be transported (including protocol requirements) is carried out in the interface unit. This allows greater flexibility as the node is unchanging irrespective of the type of data to be transported, allowing the node to be re-used in different circuits. More preferably the node directs the packet off the interconnection system to a functional unit.

More preferably, data destined for the interconnection, the interface unit can carry out the following functions: take the packet from the functional unit, ensure a correct destination module ID, head and tail bit; compare the destination module ID to the local module ID and sets a binary lane signal based on the result of this comparison; pack the moduleID, data and any high level (non bus) control signals into a flit; implement any protocol change necessary; and pass the lane signal and flit to the node using the inject protocol.

A T-junction or switch behaves in a similar way; the decision here is simply whether to route the packet down one branch or the other. This would typically be done for ranges of addresses; if the address is larger than some predefined value then the packets are routed left, otherwise they are routed right. However, more complex routing schemes could be implemented if required.

The addressing scheme can be extended to support inter-chip communication. In this case a field in the address is used to define a target chip address with, for example, 0 in this field representing a local address of the chip. When a packet arrives at the chip this field will be compared with the pre-programmed address of the chip. If they match then the field is set to zero and the local routing operates as above. If they do not match, then the packet is routed along the bus to the appropriate inter-chip interface in order to be routed towards its final destination. This scheme could be extended to allow a hierarchical addressing scheme to manage routing across systems, sub-systems, boards, groups of chips, as well as individual chips.

The system according to the present invention is not suitable for all bus-type applications. The source and destination transactors are decoupled, since there is no central arbitation point. The advantage of the approach of the present invention is that long buses (networks) can be constructed, with very high aggregrate bandwidth.

The system of the present invention is protocol agnostic. The interconnection of the present invention merely tranports data packets. Interface unit in accordance with the present invention manage all protocol specific features. This means that it easy to migrate to a new protocol, since only the interface units need to be re-designed.

The present invention also provides flexible topology and length.

The repeater blocks of the present invention allow very high clock rates in that the overall modular structure of the interconnection prevents the clock rate being limited by long wires. This simplifies the synthesis and layout. The repeater blocks not only pipeline the data as it goes downstream but they implement a flow control protocol; pipeline blockage information up the interconnection (or bus) (rather than a blocking signal being distributed globally). When blocked, a feature of this mechanism is that data compression (inherent buffering) is achieved on the bus at least double the latency figure, i.e. if latency through the repeater is one cycle then two data control flow digits (flits: the basic unit of data transfer over the interconnection of the present invention. It includes n bytes of data, as well as some side-band control signals. The flow control digit size equals the size of the bus data-path) will concatenate when it is blocked. This means that the scope of any blocking is minimised and thus reducing any queuing requirement in a functional block.

The flow of data flow control digits is managed by a flow control protocol, in conjunction with double-buffering in a store block (and repeater unit) as described previously.

The components of the interconnection of the present invention manage the tranportation of packets. Customised interface units handle protocol specific features. These typically involve packing and unpacking of control information and data, and address translation. A customised interface unit can be created to match any specific concurrency.

Many packets can be travelling along separate segments of the interconnection of the present invention simultaneously. This allows the achievable bandwidth to be much higher than the raw bandwidth of the wires (width of bus, multiplied by clock rate). If there are, for example, four adjacent on-chip blocks A, B, C and D, then A and B can communicate at the same time that C and D communicate. In this case the achievable bandwidth is twice that of broadcast-based bus.

Packets are injected (gated) onto the interconnection at each node, so that each node is allocated a certain amount of the overall bandwidth allocation (e.g. by being able to send, say 10 flow control digits within every 100 cycles). this distributed scheme controls the overall bandwidth allocation.

It is possible to keep forcing packets onto the interconnection of the present invention until it saturates. All packets will eventually be delivered. This means the interconnection can be used as a buffer with an in-built flow control mechanism.

## BRIEF DESCRIPTION OF DRAWINGS

Figure 1 is a block schematic diagram of the system incorporating the interconnection system according to an embodiment of the present invention;

Figure 2 is a block schematic diagram illustrating the initiator and target of a virtual component interface system of figure 1;

Figure 3 is a block schematic diagram of the node of the interconnection system shown in figure 1;

Figure 4 is a block schematic diagram of connection over the interconnection according to the present invention between virtual components of the system shown in figure 1;

Figure 5a is a diagram of the typical structure of the T-switch of figure 1;

Figure 5b is a diagram showing the internal connection of the T-switch of figure 5a;

Figure 6 illustrates the Module ID (interface ID) encoding of the system of figure 1;

Figure 7 illustrates handshaking signals in the interconnection system according to an embodiment of the present invention;

Figure 8 illustrates the blocking behaviour of the interconnection system of an embodiment of the present invention when occup[1:0]=01;

Figure 9 illustrates blocking for two cycles of the interconnection system according to an embodiment of the present invention;

Figure 10 illustrates virtual component interface handshake according an embodiment of the present invention;

Figure 11 illustrates a linear chip arrangement of the system according to an embodiment of the present invention;

Figure 12 is a schematic block diagram of the interconnection system of the present invention illustrating an alternative topogly;

Figure 13 is a schematic block diagram of the interconnection system of the present invention illustrating a further alternative topogly;

Figure 14 illustrates an example of a traffic handling subsystem according to an embodiment of the present invention;

Figure 15 illustrates a system for locating chips on a virtual grid according to a method of a preferred embodiment of the present invention; and

Figure 16 illustrates routing a transaction according to the method of a preferred embodiment of the present invention.

## DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

The basic mechanism for communicating data and control information between functional blocks is that blocks exchange messages using the interconnection system 100 according to the present invention. The bus system can be extended to connect blocks in a multi chip system, and the same mechanism works for blocks within a chip or blocks on different chips.

An example of a system 100 incorporating the interconnection system 110 according to an embodiment of the present invention, as shown in figure 1, comprises a plurality of reusable on-chip functional blocks or virtual component blocks 105a, 105b and 105c. These functional units interface to the interconnection and can be fixed. They can be re-used at various levels of abstraction (eg. RTL, gate level, GDSII layout data) in different circuit design. The topology can be fixed once the size, aspect ratio and the location of the I/O's to the interconnection are known. Each on-chip functional unit 105a, 105b, 105c are connected to the interconnection system 110 via its interface unit. The interface unit handles address decoding and protocol translation. The on-chip functional block 105a, for example, is connected to the interconnection system 110 via an associated virtual component interface initiator 115a and peripheral virtual component interface initiator 120a.

The on-chip functional block 105b, for example, is connected to the interconnection system 110 via an associated virtual component interface target 125b and peripheral virtual component interface target 130b. The on-chip functional block 105c, for example, is connected to the interconnection system 110 via an associated virtual component interface initiator 115c and peripheral virtual component interface target 130c. The associated initiators and targets for each on-chip functional block shown in figure 1 are purely illustrative and may vary depending on the associated block requirements. A functional block may have a number of connections to the interconnection system. Each connection has an advanced virtual component interface (extensions forming a superset of basic virtual component interface. This is the protocol used for the main data interfaces in the system of the present invention) or peripheral virtual component interface interface (low bandwidth interface allowing atomic operations, mainly used in the present invention for control register access).

One currently accepted protocol for connecting such on-chip functional units as shown in figure 1 to a system interconnection according to the embodiment of the present invention is virtual component interface. Virtual component interface is an OCB standard interface to communicate between a bus and/or virtual component, which is independent of any specific bus or virtual component protocol.

There are three types of virtual component interfaces, peripheral virtual component interface 120a, 130b, 130c, basic virtual component interface and advanced virtual component interface. The basic virtual component interface is a wider, higher bandwidth interface than the peripheral virtual component interface. The basic virtual component interface allows split transactions. Split transaction are where the request for data and the response are decoupled, so that a request for data does not need to wait for the response to be returned before initiating further transactions. Advanced virtual component interface is a superset of basic virtual component interface. Advanced virtual component interface and peripheral virtual component interface have been adopted in the system according to the embodiment of the present invention.

The advanced virtual component interface unit comprises a target and initiator. The target and initiator are virtual components that send request packets and receive response packets. The initiator is the agent that initiates transactions, for example, DMA (or EPU on F150).

As shown in figure 2, an interface unit that initiates a read or write transaction is called an initiator 210 (issues a request 220), while an interface that receives the transaction is called the target 230 (responds to a request 240). This is the standard virtual component terminology.

Communication between each no-chip functional block 105a, 105b and 105c and its associated initiators and targets are made using virtual component interface protocol. Each initiator 115a, 120a, 115c and target 125b, 130b, 130c are connected to a unique node 135, 140, 145, 150, 155 and 160. Communication between each initiator 115a, 120a, 115c and target 125b, 130b, 130c uses the protocol in accordance with the embodiment of the present invention and as described in more detail below.

The interconnection system 110 according an embodiment of the present invention comprises three separate buses 165, 170 and 175. The RTL components have parameterisable widths, so these may be three instances of different width. An example might be a 64-bit wide peripheral virtual component bus 170 (32 bit address + 32 data bits), a 128-bit advanced virtual component interface bus 165, and a 256-bit advanced virtual component interface bus 175. Although three separate buses are illustrated here, it is appreciated that the interconnection system of the present invention may incorporate any number of separate buses.

At regular intervals along the bus length a repeater unit 180 may be inserted for all the buses 165, 170 and 175. There is no restriction on the length of the buses 165, 170 and 175. Variations in the length of the buses 165, 170 and 175 would merely require an increased number of repeater units 180. Repeater units would of course only be required when the timing contraints between two nodes cannot be met due to the length of wire of the interconnection.

For complex topologies, T-switches (3-way connectors or the like) 185 can be provided. The interconnection system of the present invention can be used in any topology but care should be taken when the topology contains loops as deadlock may result.

Data is transferred on the interconnection network of the present invention in packets. The packets may be of any length that is a multiple of the data-path width. The nodes 135, 140, 145, 150, 155 and 160 according to the present invention used to create the interconnection network (node and T-switch) all have registers on the data-path(s).

Each interface unit is connected to a node within the interconnection system itself, and therefore to one particular lane of the bus. Connections may be of initiator or target type, but not both - following from the conventions of virtual component interface. In practise every block is likely to have a peripheral virtual component interface target interface for configuration and control.

The bus components according to the embodiment of the present invention use distributed arbitration, where each block in the bus system manages access to its own resources.

A node 135 according to the embodiment of the present invention is illustrated in figure 3. Each node 135, 140, 145, 150, 155 and 160 are substantially similar. Node 135 is connected to the bus 175 of figure 1. Each node comprises a first and second input store 315, 320. The first input store 315 has an input connected to a first bus lane 305. The second input store 320 has an input connected to a second bus lane 310. The output of the first input store 315 is connected to a third bus lane 306 and the output of the second input store 320 is connected to a fourth bus lane 311. Each node further comprises an inject control unit 335 and a consume control unit 325. The node may not require the consume arbitration, for example the node may have an output for each uni-directional lane but the consume handshaking retained. The input of the inject control unit 335 is connected to the output of an interface unit of the respective functional unit for that node. The outputs of the inject control unit 335 are connected to a fifth bus lane 307 and sixth bus lane 312. The input of the consume control unit 325 is connected to the output of a multiplexer 321. The inputs of the multiplexer 321 are connected to the fourth bus lane 311 and the third bus lane 306. The output of the consume control unit 325 is connected to a bus 330 which is connected to the interface unit of the respective functional unit for that node. The fifth bus lane 307 and the third bus lane 306 are connected to the inputs of a multiplexer 308. The output of the multiplexer 308 is connected to the first bus lane 305. The fourth bus lane 311 and the sixth bus lane 312 are connected to the inputs of a multiplexer 313. The output of the multiplexer 313 is connected to the second bus lane 310.

The nodes are the connection points where data leaves or enters the bus. It also forms part of the transport medium. The node forms part of the bus lane which it connects to, including both directions of data path. The node conveys data on the lane to which it connects, with one cycle of latency when not blocked. It also allows the connecting functional block to inject and consume data in either direction, via its interface unit. Arbitration of injected or passing data is performed entirely within the node.Internally, bus 175 consists of a first lane 305 and a second lane 310. The first and seond lanes 305 and 310 are physically separate uni-directional buses that are multiplexed and de-multiplexed to the same interfaces within the node 135. As illustrated in figure 3, the direction of data flow of the first lane 305 is in the opposite direction to that of the second lane 310. Each lane 305 and 310 has a lane number. The lane number is a parameter that is passed from the interface unit to the node to determine which lane (and hence which direction) each packet is sent to. Of course it is appreciated that the direction of the data flow of the first and second lanes 305 and 310 can be in the same direction. This would be desirable if the blocks transacting on the bus only need to send packets in one direction.

The node 135 is capable of concurrently receiving and injecting data on the same bus lane. At the same time it is possible to pass data through on the other lane. Each uni-directional lane 305, 310 carries a separate stream 306, 307, 311, 312 of data. These streams 306, 307, 311, 312 are multiplexed together at the point 321 where data leaves the node 135 into the on-chip module 105a (not shown here) via the interface unit 115a and

120a (not shown here). The data streams 306, 307, 311, 312 are demultiplexed from the on-chip block 105a onto the bus lanes 305 and 310 in order to place data on the interconnection 110.

This is an example of local arbitration, where competition for resources is resolved in the block 105a where those resources reside. In this case, it is competition for access to bus lanes, and for access to the single lane coming off the bus. This approach of using local arbitration is used throughout the interconnection system, and is key to its scalability. An alternative would be that both output buses come from the node to the functional unit and then the arbitration mux would not be needed.

Each lane can independently block or pass data through. Data can be consumed from one lane at a time, and injected on one lane at the same time. Concurrent inject and consume on the same lane is also permitted. Which lane each packet is injected on is determined within the interface unit.

Each input store (or register) 315 and 320 registers the data as it passes from node to node. Each store 315, 320 contains two flit-wide registers. When there is no competition for bus resources, only one of the registers is used. When the bus blocks, both registers are then used. It also implements the 'block on header' feature. This is needed to allow packets to be blocked at the header flit so that a new packet can be injected onto the bus.

The output interface unit 321, 325 multiplexes both bus lanes 305, 310 onto one lane 330 that feeds into the on-chip functional unit 105a via the interface unit which is connected to the node 135. The output interface unit 321, 325 also performs an arbitration function, granting one lane access to the on-chip functional unit, while blocking the other. Each node also comprises an input interface unit 335. The input interface unit 335 performs de-multiplexing of packets onto one of the bus lanes 305, 310. It also performs an arbitration function, blocking the packet that is being input until the requested lane is available.

A plurality of repeater units 180 are provided at intervals along the length of the interconnection 110. Each repeater unit 180 is used to introduce extra registers on the data path. It adds an extra cycle of latency, but is only used where there is a difficulty meeting timing constraints. Each repeater unit 180 comprises a store similar to the store unit of the nodes. The store unit merely passes data onwards, and implements blocking behaviour. There is no switching carried out in the repeater unit. The repeater block allows for more freedom in chip layout. For example, it allows long length of wires between nodes or where a block has a single node to connect to a single lane, repeaters may be inserted into the other lanes in order to produce uniform timing characteristics over all lanes. There may be more than one repeater between two nodes.

The system according to the embodiment of the present invention is protocol agnostic, that is to say, the data-transport blocks such as the nodes 135, 140, 145, 150, 155, 160, repeater units 180 and T-switch 185 simply route data packets from a source interface to a destination interface. Each packet will contain control information and data. The packing and unpacking of this information is performed in the interface units 115a, 120a, 125b, 130b, 115c, 130c. In respect of the preferred embodiment, these interface units are virtual component interfaces, but it is appreciated that any other protocol could be supported by creating customised interface units.

A large on-chip block may have several interfaces to the same bus.

The target and initiator 115a, 120a, 125b, 130b, 115c, 130c of the interface units perform conversion between the advanced virtual component interface and bus protocols in the initiator and from the bus to advanced virtual component interface in the target. The protocol is an asynchronous handshake on the advanced virtual component interface side illustrated in figure 10. The interface unit initiator comprises a send path. This path performs conversion between the advanced virtual component interface communication protocol to the bus protocol. It extracts a destination module ID or interface ID. Also, a block may be connected to several buses, with a different module (interface) ID on each bus address, packs it into the correct part of the packet, and uses the module ID in conjunction with a hardwired routing table to generate a lane number (e.g. 1 for right, 0 for left). The initiator blocks the data at the advanced virtual component interface when it cannot be sent onto the bus. The interface unit initiator also comprises a response path. The response path receives previously requested data, converting from bus communication protocol to the virtual component interface protocol. It blocks data on the bus if the on-chip virtual component block is unable to receive it.

The interface unit target comprises a send path which receives incoming read and write requests. The target converts from bus communication protocol to advanced virtual component interface protocol. It blocks data on the bus if it cannot be accepted across the virtual component interface. The target also comprises a response path which carries read (and for verification purposes, write) requests. It converts advanced virtual component interface communication protocol to bus protocol and blocks data at the advanced virtual component interface if it cannot be sent onto the bus.

The other type of interface unit utilised in the embodiment of the present invention is a peripheral virtual component unit. The main differences between the peripheral virtual component interface and the advanced virtual component interface are the data interface of the peripheral virtual component interface is potentially narrower (up to 4 bytes) than the advanced virtual component interface and the peripheral virtual component interface is not split transaction.

The peripheral virtual component interface units perform conversion between the peripheral virtual component interface and bus protocols in the initiator, and from the bus protocol to peripheral virtual component interface protocol in the target. The protocol is an asynchronous handshake on the peripheral virtual component interface side.

The interface unit initiator comprises a send path. It generates destination module ID and the transport lane number from memory address. The initiator blocks the data at the peripheral virtual component interface when it cannot be sent onto the bus. The initiator also comprises a response path. This path receives previously requested data, converting from bus communication protocol to the peripheral virtual component interface protocol. It also blocks data on the bus if the on-chip block (virtual component block) is unable to receive it.

The peripheral virtual component interface unit target comprises a send path which receives incoming read and write requests. It blocks data on the bus if it cannot be accepted across the virtual component interface. The target also comprises a respeonse path which carries read (and for verification purposes, write) requests. It converts peripheral virtual component interface communication protocol to bus protocol and blocks data at the virtual component interface if it cannot be sent onto the bus.

The peripheral virtual component interface initiator may comprise a combined initiator and target. This is so that the debug registers (for example) of an initiator can be read from.

With reference to figure 4, the virtual component (on-chip) blocks can be connected to each other over the interconnection system according to the present invention. A first virtual component (on-chip) block 425 is connected point to point to an interface unit target 430. The interface unit target 430 presents a virtual component initiator interface 440 to the virtual component target 445 of on-chip block 425. The interface unit target 430 uses a bus protocol conversion unit 448 to interface to the bus interconnect 450. The interface unit initiator 460 presents a target interface 470 to the initiator 457 of the second on-chip block 455 and, again, uses a bus protocol conversion unit 468 on the other side.

The T-switch 185 of figure 1 is a block that joins 3 nodes, allowing more complex interconnects than simple linear ones. At each input port the interface ID of each packet is decoded and translated into a single bit, which represents the two possible outgoing ports. A hardwired table inside the T-Switch performs this decoding. There is one such table for each input port on the T-Switch. Arbitration takes place for the output ports if there is a conflict. The winner may send the current packet, but must yield when the packet has been sent. Figures 5a and 5b show an example of the structure of a T-switch.

The T-switch comprises three sets of input/output ports 505, 510, 515 connected to each pair of unidirectional bus lanes 520, 525, 530. Within the T-switch, a T-junction 535, 540, 545 is provided for each pair of bus lanes 520, 525, 530 such that an incoming bus 520 coming into an input port 515 can be output via output port 505 or 510, for example.

Packets do not change lanes at any point on the bus, so the T-switch can be viewed as a set of n 3-way switches, where n is the number of uni-directional bus lanes. The T-switch 185 comprises a lane selection unit. The lane selction unit takes in module ID of incoming packets and produces a 1-bit result corresponding to the two possible output ports on the switch. The T-switch also comprises a store block on each input lane. Each store block stores data flow control digits and allows them to block in place if the output port is temporarily unable to receive. It also performs a 'block on header' function, which allows switching to occur at the packet level (rather than the flow control digit level). The T-switch also includes an arbiter for arbitration between requests to use output ports.

During initialisation, the interconnection system according to the embodiment of the present invention powers up into a usable state. Routing information is hardcoded into the bus components. A destination module interface ID (mod ID) for example as illustrated in figure 6 is all that is required to route a packet to another node. In order for that node to return a response packet, it must have been sent the module interface ID of the sender.

There may be more than one interconnection in a processing system. On each bus, every interface (which includes an inject and consume port) has a unique ID. These ID's are hard-coded at silicon compile-time.

Units attached to the bus (on-chip blocks) are free to start communicating straight after reset. The interafce unit will hold off communications (by not acknowledging them) until it is ready to begin operation.

The interconnection system according to the present invention has an internal protocol that is used throughout. At the interfaces to the on-chip blocks this may be converted to some other protocol, for example virtual component interface as described above. The internal protocol will be referred to as the bus *protocol.* This bus protocol allows single cycle latency for packets travelling along the bus when there is no contention for resources, and to allow packets to block in place when contention occurs.

The bus protocol is used for all internal (non interface/virtual component interface) data transfers. It consists of five signals: *occup* 705, *head* 710, *tail* 715, *data* 720 and *valid* 725 between a sender 735 and a receiver 730. These are shown in figure 7.

The packets consist of one or more flow control digits. On each cycle that the sender asserts the valid signal, the receiver must accept the data on the next positive clock edge.

The receiver 730 informs the sender 735 about its current status using the occup signal 705. This is a two-bit wide signal.

| Occup [1:0] | Meaning |
|---|---|
| 00 | Receiver is empty – can send data. |
| 01 | Receiver has one flow control digit – if |

| | |
|---|---|
| | sending a flow control digit on this cycle, don't send a flow control digit on the next cycle. |
| 10 | The Receiver is full. Don't send any flow control digits until Occup decreases. |
| 11 | Unused. |

**Table I: Occup signal values and their meaning.**

The occup signal 705 tells the sender 735 if and when it is able to send data. When the sender 735 is allowed to transmit a data flow control digit, it is qualified with a valid signal 725.

The first flow control digit in each packet is marked by *head* = *'1'*. The last flow control digit is marked by *tail* = *'1'*. A single flow control digit packet has signals *head* = *tail* =*valid* = *'1'*. Each node and T-Switch use these signals to perform switching at the packet level.

Figure 8 shows an example of blocking behaviour on the interconnect system according to an embodiment of the present invention. The occup signal is set to '1', meaning 'if sending a flow control digit this cycle, don't send one on the next cycle'.

Figure 9 shows an example of the blocking mechanism more completely. The occup signal is set to 01 (binary), then to 10 (binary). The sender can resume transmitting flow control digits when the occup signal is set back to 01 – at that point it is not currently sending a flow control digit, so it is able to send one on the next cycle.

The protocol at the boundary between the node and the interface unit is different from that just described, and is similar to that used by the virtual component interface. At the sending and receiving interfaces, there is a *val* and an *ack* signal. When *val* = *ack* = *1*, a flow control digit is exchanged for the inject protocol. The consume (bus output) protocol is different to the inject protocol but is the minimum logic to allow registered outputs (and thus simplifies synthesis and integration into a system on chip). The consume protocol is defined as: on the rising clock edge, data is blocked on the next clock edge if CON_BLOCK =1; on the rising clock edge, data is unblocked on the next clock edge if CON_BLOCK = 0; CON_BLOCK is the flow control (blocking) signal from the functional unit.

Of course the protocols at this interface can be varied and not effect the overall operation of the bus.

The difference between this and virtual component interface is that the *ack* signal is high by default, and is only asserted low on a cycle when data cannot be received. Without this restriction, the node would need additional banks of registers.

The bus protocol allows exchange packets consisting of one or more flow control digits. Eight bits in the upper part of the first packet carry the destination module ID, and are used by the bus system to deliver the packet. The top 2 bits are also used for internal bus purposes. In all other bit fields, the packing of the flow control digits is independent of the bus system.

At each interface unit, virtual component interface protocol is used. The interface control and data fields are packed into bus flow control digits by the sending interface and then unpacked at the receiving interface unit. The main, high-bandwidth, interface to the bus uses the advanced virtual component interface. All features of the advanced virtual component interface are implemented, with the exception of those used to optimise the internal operation of an OCB.

The virtual component interface protocol uses an asynchronous handshake as shown in Figure 8. Data is valid when *VAL = ACK = 1.* The bus interface converts data and control information from the virtual component interface protocol to the bus internal communication protocol.

The bus system does not distinguish between control information and data. Instead, the control bits and data are packed up into packets and sent to the destination interface unit, where they are unpacked and separated back into data and control.

Although in the preferred embodiment, virtual component interface compliant interface units are utilised, it is appreciated that different interface units may be used instead (e.g. ARM AMBA compliant interfaces).

Table II shows the fields within the data flow control digits that are used by the interconnection system according to an embdoiemnt of the present invention. All other information in the flow control digits is simply transported by the bus. The encoding and decoding is performed by the interface units. The interface units also insert the head and tail bits into the flow control digits, and insert the MOD ID in the correct bit fields.

| **Name** | **Bit** | **Comments** |
|---|---|---|
| Head | FLOW CONTROL DIGIT_WIDTH - 1 | Set to '1' to indicate first flow control digit of packet. |
| Tail | FLOW CONTROL DIGIT_WIDTH - 2 | Set to'1' to indicate last flow control digit of packet. |
| Mod ID | FLOW CONTROL DIGIT_WIDTH –3: FLOW CONTROL DIGIT_WIDTH -10 | ID of interface to which packet is to be sent. Virtual component interface calls this MOD ID. It is really an interface ID, since a large functional unit could have multiple bus interfaces, in which case, it is necessary to distinguish between |

| | | |
|---|---|---|
| | | them. |

**Table II: Specific fields.**

The advanced virtual component interface packet types are read request, write request and read response. A read request is a single flow control digit packet and all of the relevant virtual component interface control fields are packed into the flow control digit. A write request consists of two or more flow control digits. The first flow control digit contains virtual component interface control information (e.g. address). The subsequent flow control digits contain data and byte enables. A read response consists of one or more flow control digits. The first and subsequent flow control digits all contain data plus virtual component interface response fields (e.g. *RSCRID, RTRDID and RPKTID*).

An example mapping of the advanced virtual component interface onto packets is now described. The example is for a bus with 128-bit wide data paths. It should be noted that the nodes extract the destination module ID from bits 159:152 in the first flow control digit of each packet. In the case of read response packets this corresponds with the virtual component interface RSCRID field.

| AVCI/BVCI Signal Name | WIDTH | Bits | Header Flow control digit Only? | Read Responses Only? | Direction | Comments |
|---|---|---|---|---|---|---|
| CLOCK | 1 | - | - | - | IA | |
| RESETN | 1 | - | - | - | IA | |
| CMDACK | 1 | - | - | - | TI | Handshake signal. |
| CMDVAL | 1 | - | - | - | IT | Handshake signal. |
| WDATA | 128 | 127:0 | | | IT | Only for write requests. |
| BE | 16 | 143:128 | | | IT | Only for write requests. |
| ADDRESS | 64 | 63:0 | ŏ | - | IT | |
| CFIXED | 1 | 64 | ŏ | | IT | |
| CLEN | 8 | 72:65 | ŏ | | IT | ***needs update*** |
| CMD | 2 | 75:74 | ŏ | | IT | |
| CONTIG | 1 | 76 | ŏ | | IT | |
| EOP | 1 | - | - | - | IT | Handshake signal. |
| CONST | 1 | 77 | ŏ | | IT | |
| PLEN | 9 | 86:78 | ŏ | | IT | |
| WRAP | 1 | 87 | ŏ | | IT | |
| RSPACK | 1 | - | - | - | IT | Handshake signal. |
| RSPVAL | 1 | - | - | - | TI | Handshake<br>signal. |
| RDATA | 128 | 127:0 | | ŏ | TI | Only for read responses. |

| | | | | | | |
|---|---|---|---|---|---|---|
| **REOP** | 1 | - | - | - | TI | Handshake signal. |
| **RERROR** | 2 | 143:142 | | ŏ | TI | Only for read responses. |
| **DEFD** | 1 | 88 | ŏ | | IT | |
| **WRPLEN** | 5 | 93:89 | ŏ | | IT | |
| **RFLAG** | 4 | 141:138 | | ŏ | TI | Only for read responses. |
| **SCRID** | 8 | 151:144 | ŏ | | IT | |
| **TRDID** | 2 | 95:94 | ŏ | | IT | |
| **PKTID** | 8 | 103:96 | ŏ | | IT | |
| **RSCRID** | 8 | 159:152 | | ŏ | TI | Only for read responses. |
| **RTRDID** | 2 | 137:136 | | ŏ | TI | Only for read responses. |
| **RPKTID** | 8 | 135:128 | | ŏ | TI | Only for read responses. |
| IT = Initiator to target<br>TI = Target to initiator<br>IA = Input to all devices | | | ŏ Field exists in these flow control digits.<br>Field not included in these flow control digits.<br>- Not applicable – Signal that does not cause data to be sent (such as handshake signal). | | | |

**Table III: Possible Virtual Component Interface fields for 128 bit wide bus**

Peripheral virtual component interface burst-mode read and write transactions are not supported over the bus, as these cannot be efficiently implemented. For this reason, the peripheral virtual component interface EOP signal should be fixed at logic '1'. Any additional processing unit or extenal units can be attached to the bus, but the EOP signal should again be fixed at logic '1'. With this change, the new unit should work normally.

The read request type is a single flow control digit packet carrying the 32-bit address of the data to be read. The read response is a single flow control digit response containing the requested 32 bits of data. The write request is a single flow control digit packet containing the 32-bit address of the location to be written, plus the 32 bits of data, and 4 bits of byte enable. The write response prevents a target responding to a write request in the same way that it would to a read request.

With all of the additional signals, 32 bit (data) peripheral virtual component interface occupies 69 bits on the bus.

| PVCI Signal Name | Bit Fields | Read Request | Read Response | Write Request | Comments |
|---|---|---|---|---|---|
| CLOCK | - | - | - | - | System signal. |
| RESETN | - | - | - | - | System signal. |
| VAL | - | - | - | - | Handshake signal. |
| ACK | - | - | - | - | Handshake signal. |
| EOP | - | - | - | - | Handshake signal. |
| ADDRESS | 63:32 | ŏ | | ŏ | |
| RD | 100 | ŏ | | ŏ | |
| BE | 67:64 | | | ŏ | |
| WDATA | 31:0 | | | ŏ | |
| RDATA | 31:0 | | ŏ | | |
| RERROR | 68 | | ŏ | | |

**TABLE IV**

The internal addressing mechanism of the bus is based on the assumption that all on-chip blocks in the system have a fixed 8-bit module ID.

Virtual component interface specifies the use of a single global address space. Internally the bus delivers packets based on the module ID of each block in the system. The module ID is 8 bits wide. All global addresses will contain the 8 bits module ID, and the interface unit will simply extract the destination module ID from the address. The location of the module ID bits within the address is predetermined.

The module IDs in each system are divided into groups. Each group may contain up to 16 modules. The T-switches in the system use the group ID to determine which output port to send each packet to.

Within each group, there may be up to sixteen on-chip blocks, each with a unique subID. The inclusion of only sixteen modules within each group does not restrict the bus topology. Within each linear bus section, there may be more than one group, but modules from different groups may not interleave. There may be more than sixteen modules between T-switches. The only purpose of using a group ID and sub ID is to simplify the routing tables inside the T-switch(es). If there are no T-switches being used, the numbering of modules can be arbitrary. If a linear bus topology is used and the interfaces are numbered sequentially, this may simplify lane number generation, as a comparator can be used instead of a table. However, a table may still turn out to be smaller after logic minimisation. Two interfaces on different buses can have the same mod ID (= interface ID).

An example to reduce erroneous traffic on the interconnection according to the embodiment of the present invention is described here. When packets that do not have a legal mod ID are presented to the interface unit, it will acknowledge them, but will also generate an error in the *rerror* virtual component interface field. The packet will not be sent onto the bus. The interface unit will "absorb" it and destroy it.

In the preferred embodiment the bus system blocks operate at the same clock rate as synthesisable RTL blocks in any given process. For a 0.13 µm 40G processor, the target clock rate is 400 MHz. There will be three separate buses. Each bus cpmprises a separate parameterised component. There will be a 64-bit wide peripheral virtual component interface bus connecting to all functional units on the chip. There will be two advanced virtual component interface buses, one with a 128-bit data-path (raw bandwidth 51.2 Gbits/sec on each unidirectional lane), the other with a 256-bit data-path (raw bandwidth 102.4 Gbits/sec on each unidirectional lane). Not all of this bandwidth can be fully utilised due to the overhead of control and request packets, and it is not always possible to achieve efficient packing of data into flow control digits. Some increase in bandwidth will be seen due to concurrent data transfers on the bus, but this can only be determined in system simulations.

In the embodiment, latency of packets on the bus is one cycle at the sending interface unit, one cycle per bus block (nodes and repeaters) that data passes through, one or two additional cycle(s) at the node consume unit, one cycle at receiving interface unit and *n-1* cycles, where *n* is the packet length in flow control digits. This figure gives the latency for the entire data transfer, meaning that latency increases with packet size.

It is possible to control bandwidth allocation, by introducing programmability into the injection controller in the node.

It is not possible for packets to be switched between bus lanes. Once a packet has entered the bus system, it stays on the same bus lane, until it is removed at the destination interface unit. Packets on the bus are never allowed to interleave. It is not necessary to inject new flow control digits on every clock cycle. In other words, it is possible to have gap cycles. These gaps will remain in the packet when it is inside the bus system, and will waste bandwidth. These packets will remain in the packet when it is in the bus system and unblocked, thus wasting bandwidth. If blocked then only validated data will concatenate and thus any intermediate non valid data will be removed. In addition to help minimise the number of gap packets, it is necessary to ensure that enough FIFO buffering is provided to allow the block to keep injecting flow control digits until each packet has been completely sent, or design the block in a manner that does not cause gaps to occur.

In the system according to the present invention, the length of the packets (in flow control digits) is unlimited. However, consideration should be taken with excessively large packets, as they will utilises a greater amount of the bus resource. Long packets can be more efficient than a number of shorter ones, due to the overhead of having a header flow control digit.

The interconnection system according to the present invention does not guarantee that requested data items would be returned to a module in the order in which they were requested. The block is responsible for re-ordering packets if the order matters. This is achieved using the advanced virtual component interface *pktid* field, which is used to tag and reorder outstanding transactions. It cannot be assumed that data will arrive at the on-chip block in the same order that it was requested from other blocks in the system. Where ordering is important, the receiving on-chip block must be able to re-order the packets. Failure to adhere to this rule is likely to result in system deadlock.

The interconnection system according to the present invention offers considerable flexibility in the choice of interconnect topology. However, it is currently not advisable to have loops in the topology as these will introduce the possibility of deadlock.

However, it should be possible to program routing tables in a deadlock-free manner if loops were to be used. This would require some method of proving deadlock freedom, together with software to implement the necessary checks.

A further advantage of the interconnection system according to the present invention is that saturating the bus with packets will not cause it to fail. The packets will be delivered eventually, but the average latency will increase significantly. If the congestion is reduced to below the maximum throughput, then it will return to “normal operation”.

In this repsect the following rules should be considered, namely there should be no loops in the bus topology, on-chip blocks must not depend on transactions being returned in order and where latency is important, and multiple transactors need to use the same bus segments, there should be a maximum packet size.. As mentioned above, if loops are required in the future, some deadlock prevention strategy must exist. Ideally this will include a formal proof. Further, if ordering is important, the blocks must be able to re-order the transaction. Two transactions from the same target, travelling on the same lane will be returned in the same order in which the target sent them. If requests were made to two different targets, the ordering is non-deterministic.

Most of the interconnection components of the present invention will involve straightforward RTL synthesis followed by place & route. The interface units may be incorporated into either the on-chip blocks or an area reserved for the bus, depending on the requirements of the overall design, for example there may be a lot of area free under the bus and so using this area rather than adding the functional block area would make more sense as it would reduce the overall chip area.

The interconnection system forms the basis of a system-on-chip platform. In order to accelerate the process of putting a system on-chip together, it has been proposed that the nodes contain the necessary "hooks" to handle distribution of the system clock reset signals. Looking first at an individual chip, the routing of transactions and responses between initiator and target is performed by the interface blocks that connect to the interconnection system, and any intervening T-switch elements in the system. Addressing of blocks is hardwired and geographic, and the routing information is compiled into the interface and T-switch logic at chip integration time. The platform requires some modularity at the chip level as well as the block level on chips. Therefore, knowledge of what other chips or their innards they are connected to can not be hard-coded in the chips themselves, as this may vary on different line cards.

However, with the present invention, it is possible to provide flexibility of chip arrangement with hard-wired routing information by giving each chip some simple rules and designing the topology and enumeration of the chips to support this. This has the dual benefit of simplicity and of being a natural extension to the routing mechanisms within chips themselves.

Figure 11 illustrates an example of a linear chip arrangement. Of course, it is appreciated that different topologies can be realised according to the present invention. In such a linear arrangement, it is easy to number the chips 1100(0) to (2) etc sequentially so that any block in any chip knows that a transaction must be routed “up” or “down” the interconnection 1110 to reach its destination, as indicated in the chip ID field of the physical address. This is exactly the same process as the block performs to route to another block on the same chip. In this case a 2-level decision is utilised. If in ‘present chip’ then route on Block ID, else route on Chip ID.

An alternative topology is shown in figure 12. It comprises a first bus lane 1201 and a second bus lane 1202 arranged in parallel. The first and second bus lane correspond to the interconnection system of the embodiment of the present invention. A plurality of multi threaded array processors (MTAPs) 1210 are connected across the two bus lanes 1201, 1202. An network input device 1220, a collector device 1230, a distributor device 1240, an network output device 1250 are connected to the second bus lane 1202 and a table lookup engine 1260 is connected to the first bus lane 1201. Details of operation of the devices connected to the bus lanes is not provided here.

As illustarted in figure 12, in an alternative topology, the first bus lane 1201 (256 bits wide, for example) is dedicated to fast path packet data. A second bus lane 1202 (128 bits wide, for example) is used for general non-packet data, such as table lookups, instruction fetching, external memory access, etc. Blocks accessing bus lanes 1201, 1202 use AVCI protocol. An additional bus lane may be used (not shown here for reading and writing block configuration and status registers. Blocks accessing this lane use the PVCI protocol.

More generally, where the blocks are connected to the interconnection and which lane or lanes they use can be selected. Allowance for floor planning constraints must obviously be taken into account. Blocks can have multiple bus interfaces, for example. Lane widths can be configured to meet the bandwidth requirements of the system.

Since the interconnection system of the present invention uses point-to-point connections between interfaces, and uses distributed arbitration, it is possible to have several pairs of functional blocks communicating simultaneously without any contention or interference. Traffic between blocks can only interfere if that traffic travels along a shared bus segment in the same direction. This situation can be avoided by choosing a suitable layout. Thus, bus contention can be avoided in the fast path packet flow. This is important to achieve predictable and reliable performance, and to avoid overprovisioning the interconnection.

The example above, avoids bus contention in the fast path, because the packet data flows left to right on bus lane 1201 via NIP-DIS-MTAP-COL-NOP. Since packets do not cross any bus segment more than once, there is no bus contention. There is no interference between the MTAP processors, because only one at a time is sending or receiving. Another way to avoid bus contention is to place the MTAP processors on a "spur" off the main data path, as shown in figure 13.

This topology uses a T-junction 1305 to exploit the fact that traffic going in opposite directions on the same bus segment 1300 is non-interfering. Using the T-junction block 1305 may ease the design of the bus topology to account for layout and floor planning constraints.

At the lowest (hardware) level of abstraction, the interconnection system of the present invention preferrably supports advanced virtual component interface *transactions*, which are simply variable size messages as defined in the virtual component interface standard, sent from an initiator interface to a target interface, possibly followed by a response at a later time. Because the response may be delayed, this is called a split transaction in the virtual component interface system. The network processing system architecture defines two higher levels of abstraction in the inter-block communication protocol, the *chunk* and the *abstract datagram* (frequently simply called a *datagram*). A chunk is a logical entity that represents a fairly small amount of data to be transferred from one block to another. An abstract datagram is a logical entity that represents the natural unit of data for the application. In network processing applications, abstract datagrams almost always correspond to network datagrams or packets. The distinction is made to allow for using the architecture blocks in other applications besides networking. Chunks are somewhat analogous to CSIX C-frames, and are used for similar purposes, that is, to have a convenient, small unit of data transfer. Chunks have a defined maximum size, typically about 512 bytes, while datagrams can be much larger, typically up to 9K bytes; the exact size limits are configurable. When a datagram needs to be transferred from one block to another, the actual transfer is done by sending a sequence of chunks. The chunks are packaged within a series of AVCI transactions at the bus interface.

The system addressing scheme according to the embodiment of the present invention will now be described in more detail. The system according to an embodiment of the present invention may span a subsystem that is implemented in more than one chip.

Looking first at an individual chip, the routing of transactions and responses between initiators and targets is performed by the interface blocks that connect to the interconnection itself, and the intervening T-switch elements in the interconnection. Addressing according to an embodiment of the present invention of the blocks is hardwired and geographic, and the routing information is compiled into the interface units, T-switch and node elements logic at chip integration.

The interface ID, occupies the upper part of the physical 64 bit address, the lower bits being the offset within the block. Additional physical bits are reserved for the chip ID to support multi-chip expanses.

The platform according to the embodiment of the present invention requires some modularity at the chip level as well as the block level on chips, knowledge of what other chips or their innards they are connected to can not be hard-coded, as this may vary on different line cards. This prevents the use of the same hard-wired bus routing information scheme as exists in the interface units for transactions within one chip.

However, it is possible to provide flexibility of chip arrangement with hard-wired routing information by giving each chip some simple rules and designing the topology and enumeration of chips to support this. This has the dual benefits of simplicity and of being a natural extension to the routing mechanisms within the chips themselves.

An example of a traffic handler subsystem in which the packet queue memory is implemented around two memory hub chips is shown in figures 14 and 15.

In the example, the four chips have four connections to other chips. This results in possible ambiguities about the route that a transaction takes from one chip to the next. Therefore, it is necessary to control the flow of transactions by configuring the hardware and software appropriately, but without having to include programmable routing functions in the interconnection.

This is achieved by making the chip ID an x,y coordinate instead of a single number. For example, chip ID for chip 1401 may be 4,2, chip 1403 may be 5,1, chip 1404 may be 5,3 and chip 1402 may be 6,2. Simple, hardwired rules are applied on about how to route the next hop of a transaction destined for another chip. Thus, locating the chips on a virtual “grid” such that the local rules produce the transaction flows desired. The grid can be “distorted” by leaving gaps or dislocations to achieve the desired effect.

Each chip has complete knowledge of itself, including how many external ports it has and their assignments to N,S,E,W compass points. This knowledge is wired into the inetrface units and T-switches. A chip has no knowledge at all of what other chips are connected to it, or their x,y coordinates.

The local rules at each chip are this:

1. A transaction is routed out on the interface that forms the least angle with its relative grid location.

2. In the event of a tie, N-S interfaces are favoured over E-W.

Applying these rules to the four chip example above, transactions along the main horizontal axis through Arrivals & Dispatch chips 1401, 1402 are simply routed using up/down on the x coordinate, with y=2.

Transactions from Arrivals or Dispatch 1401, 1402 to one of the memory hubs 1403, 1404 have an angle of 45 degrees, and the 2nd rule applies to route these N-S and not E-W.

Responses from memory hubs 1403, 1404 to any other chip have a clear E/W choice because no other chip has x=5.

The conjecture is that there is no chip topology or transaction flow that cannot be expressed by suitable choice of chip coordinates and application of the above rules.

Although a preferred embodiment of the method and system of the present invention has been illustrated in the accompanying drawings and described in the forgoing detailed description, it will be understood that the invention is not limited to the embodiment disclosed, but is capable of numerous variations, modifications without departing from the scope of the invention as set out in the following claims.

**CLAIMS:**

1. An interconnection system for connecting a plurality of functional units, the interconnection system comprising a plurality of nodes, each node communicating with a functional unit, the interconnection system transporting a plurality of data packets between functional units, each data packet has routing information associated therewith to enable a node to direct the data packet via the interconnection system.

2. An interconnection system for interconnecting a plurality of functional units and transporting a plurality of data packets, the interconnection system comprising a plurality of nodes, each node communicating with a functional unit wherein, during transportation of the data packets between a first node and a second node, only the portion of the interconnection system between the first node and the second node is activated.

3. An interconnection system for interconnecting a plurality of functional units, each functional unit connected to the interconnection system via an interface unit, the interconnection system comprising a plurality of nodes, the interconnection system

transporting a plurality of data packets wherein each interface unit translate the protocol for transporting the data packets and the protocol of the functional units.

4. An interconnection system for interconnecting a plurality of functional units and transporting a plurality of data packets between the functional units wherein arbitration is distributed to each functional unit.

5. An interconnection system according to any one of claims 2 to 4, wherein each data packet has routing information associated therewith to enable a node to direct the data packet via the interconnection system.

6. An interconnection system according to any one of claims 1, 3 or 4, wherein, during transportation of a data packet between a first node and a second node, only the portion of the interconnection system between the first node and the second node is activated.

7. An interconnection system according to claims 1, 2 or 4, wherein the interconnection system is protocol agnostic.

8. An interconnection system according to any one of claims 1 to 3, wherein arbitration is distributed between the functional units.

9. An interconnection system according to any one of the preceding claims further comprising a plurality of repeater units spaced along the interconnection at predetermined distances such that the data packets are transported between consecutive repeater units and/or nodes in a single clock cycle.

10. An interconnection system according to claim 9, wherein the data packets are pipelined between the nodes and/or repeater units

11. An interconnection system according to claim 9 or 10, wherein each repeater unit comprises means to compress data upon blockage of the interconnection.

12. An interconnection system according to any one the preceding claims, wherein the routing information includes the x,y coordinates of the destination.

13. An interconnection system according to any one of the preceding claims, wherein the clocking along the length of the interconnection system is distributed.

14. An interconnection system according to any one of the preceding

claims, wherein each node comprises an input buffer, inject control and/or consume control.

15.An interconnection system according to claim 14, wherein each node can inject and output data at the same time.

16.An interconnection system according to any one of the preceding claims, wherein the interconnection system comprises a plurality of buses.

17.An interconnection system according to claim 16, wherein each node is connected to at least one of the plurality of buses.

18.An interconnection system according to claim 16 or 17, wherein at least a part of each bus comprises a pair of unidirectional bus lanes.

19.An interconnection system according to claim 15, wherein data is transported on each bus lane in an opposite direction.

20.An interconnection system according to any one of the preceding claims, further comprising at least one T-switch, the T-switch determining the direction to transport the data packets from the routing information associated with each data packet.

21. An interconnection system according to any one of the preceding claims, wherein delivery of the data packet is guaranteed.

22. A method for routing data packet between functional units, each data packet has routing information associated therewith, the method comprising the steps of:

(a) reading the routing information;

(b) determining the direction to transport the data packet from the routing information; and

(c) transporting the data packet in the direction determined in step (b).

23. A processing system incorporating the interconnection system according to any one of claims 1 to 21.

24. A system according to claim 23, wherein each functional unit is connected to a node via an interface unit.

25. A system according to claim 24, wherein each interface unit comprises means to set the protocol for data to be transported to the interconnection system and to be received from the interconnection system.

26.A system according to any one of claims 24 to 25, wherein the functional units access the interconnection system using distributed arbitration.

27.A system according to any one of claims 24 to 26, wherein each functional unit comprises a reusable system on chip functional unit.

28. An integrated circuit incorporating the interconnection system according to any one of claim 1 to 21.

29.An integrated system comprising a plurality of chips, each chip incorporating the interconnection system according to any one of claims 1 to 21, wherein the interconnection system interconnects the plurality of chips.

30.A method for transporting a plurality of data packets via an interconnection system, the interconnection system comprising a plurality of nodes, the method comprising the steps of:

transporting a data packet between a first node and a second node; and

during transportation, only activating the portion of the interconnection system between the first node and the second node.

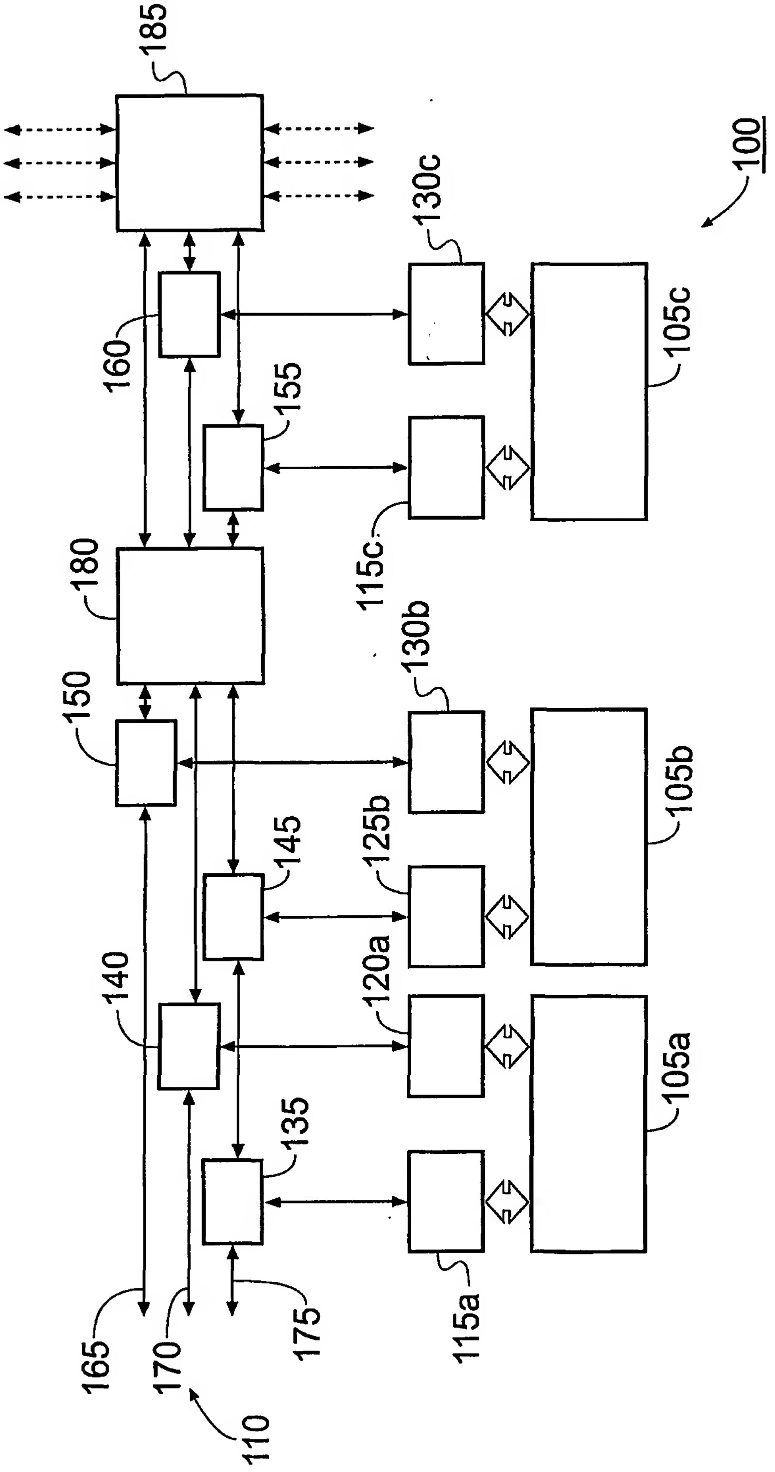

185
100
130c
160
155
105c
180
115c
130b
150
145
125b
105b
140
120a
135
105a
115a
165
170
175
110

Fig. 1

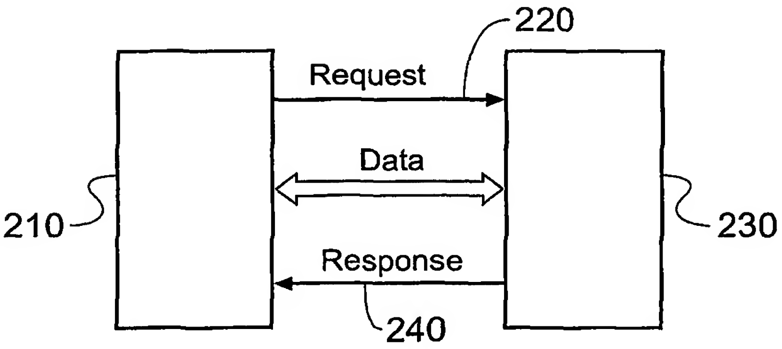

220
Request
Data
210
230
Response
240

Fig. 2

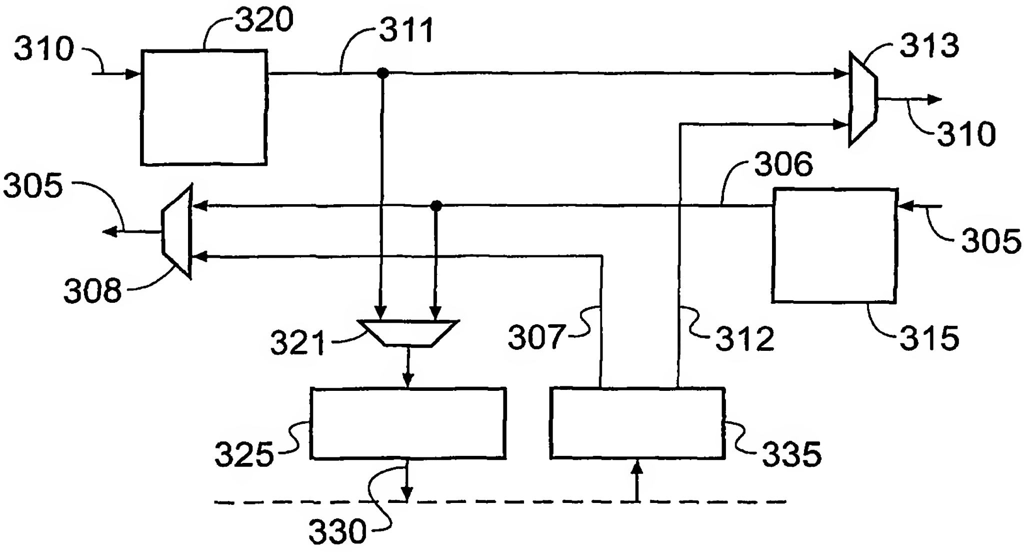

320
311
313
310
310
310
306
305
305
308
321
307
312
315
325
335
330

Fig. 3

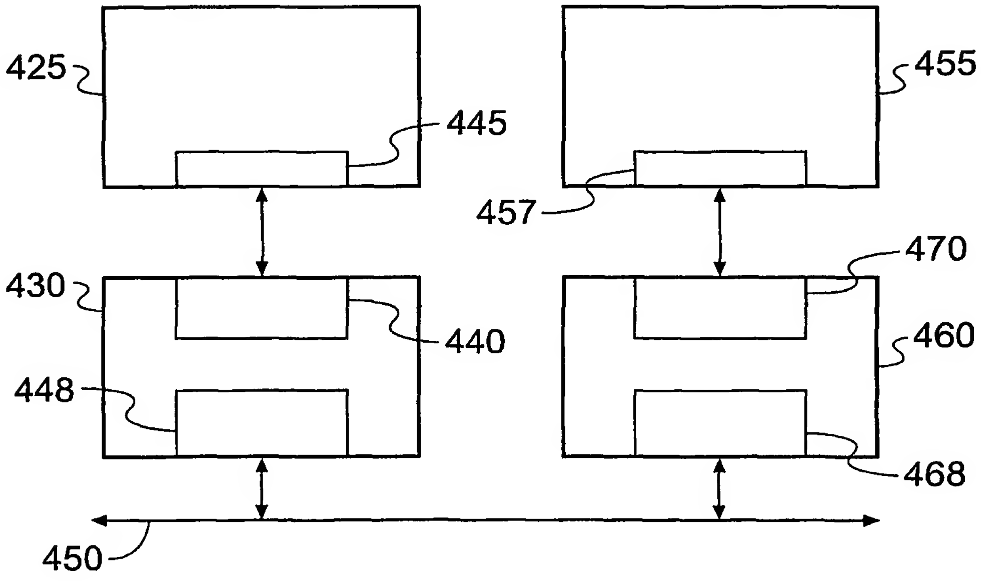


Fig. 4

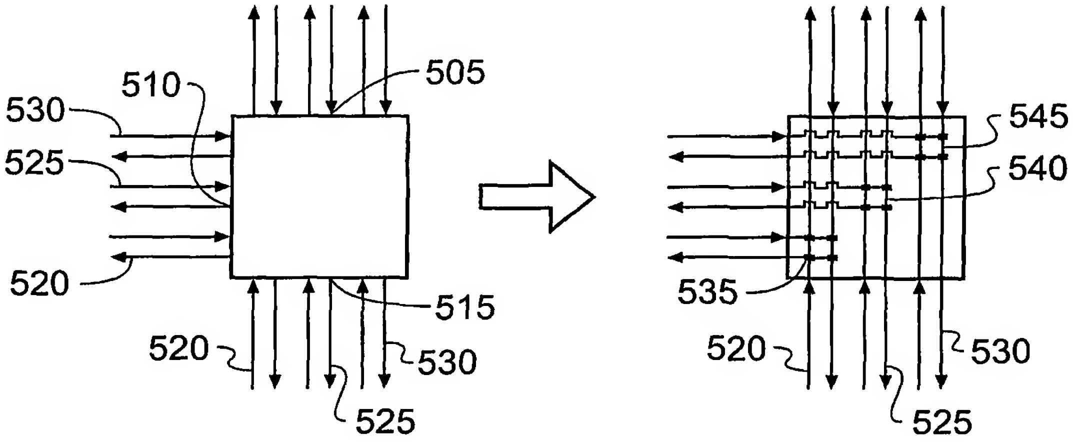


Fig. 5a

Fig. 5b

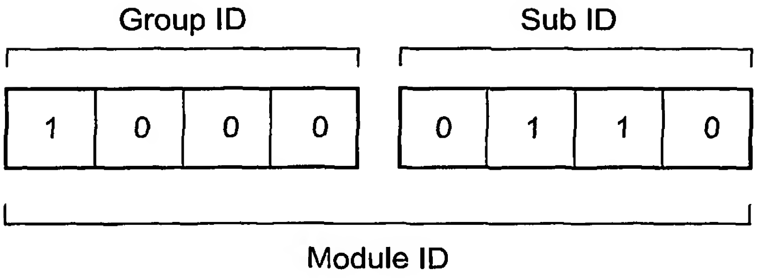


Fig. 6

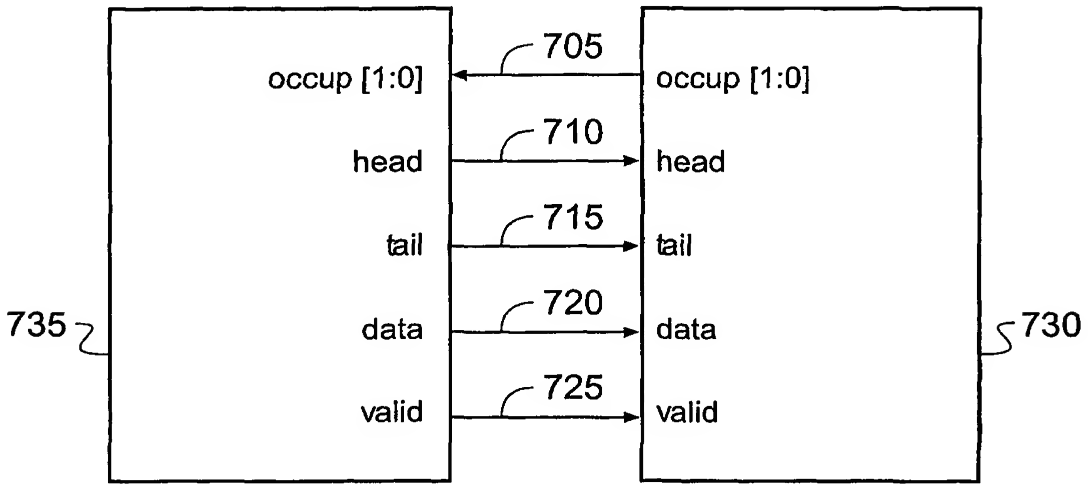

occup [1:0]
head
tail
data
valid
705
710
715
720
725
occup [1:0]
head
tail
data
valid
735
730

Fig. 7

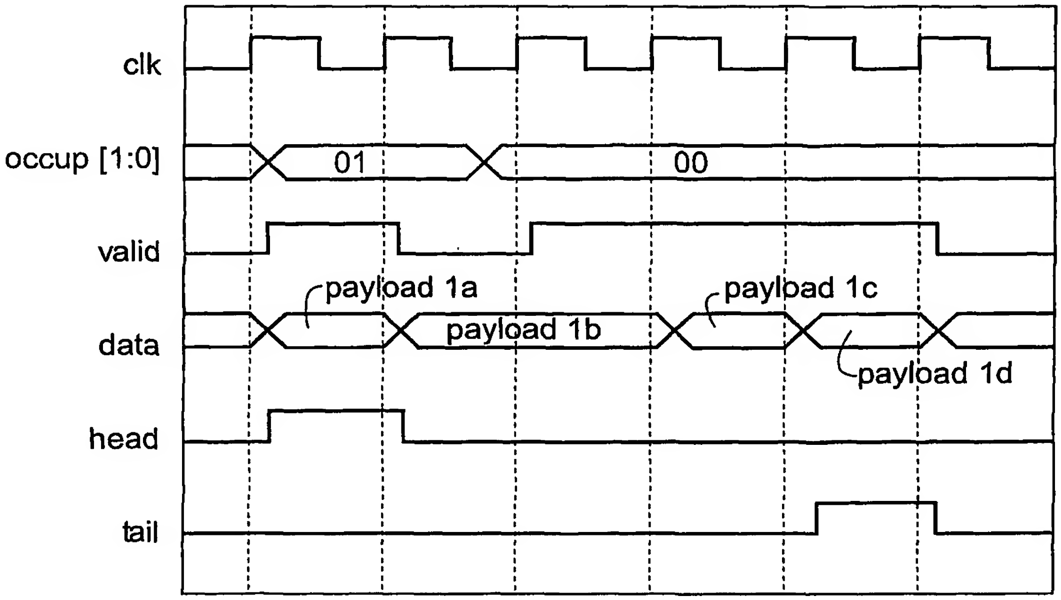

clk
occup [1:0]
01
00
valid
payload 1a
payload 1c
data
payload 1b
payload 1d
head
tail

Fig. 8

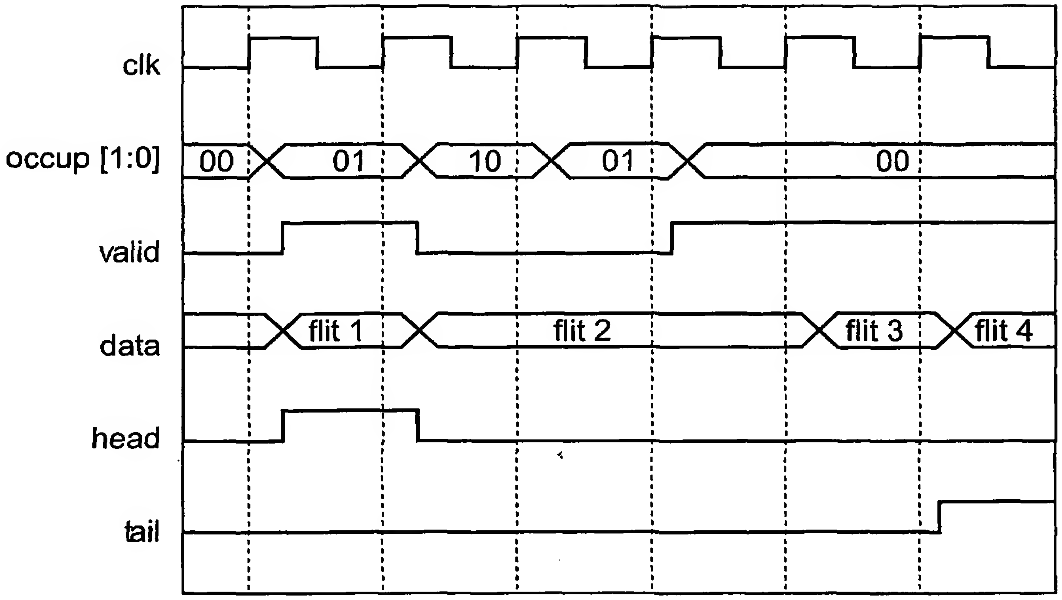

clk
occup [1:0]
00
01
10
01
00
valid
data
flit 1
flit 2
flit 3
flit 4
head
tail

Fig. 9

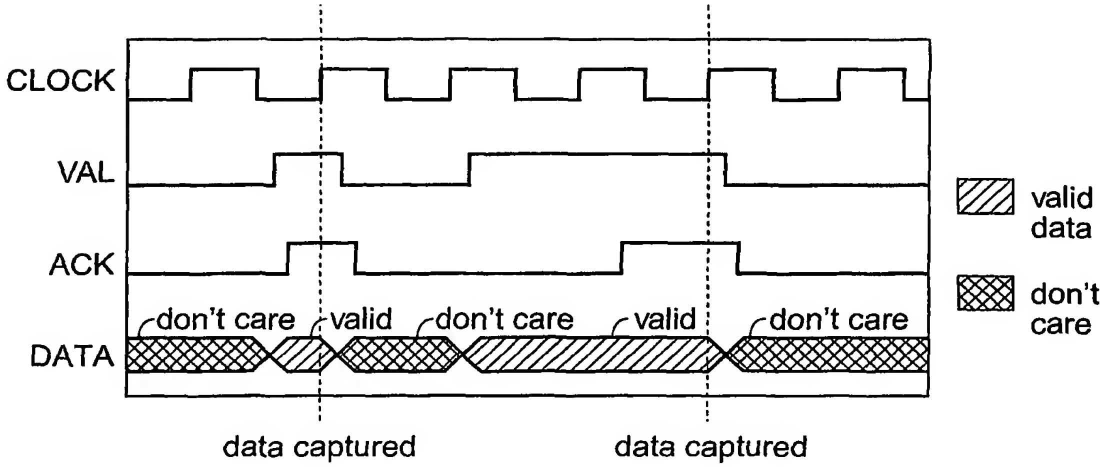

CLOCK
VAL
ACK
DATA
don't care
valid
don't care
valid
don't care
valid data
don't care
data captured
data captured

Fig. 10

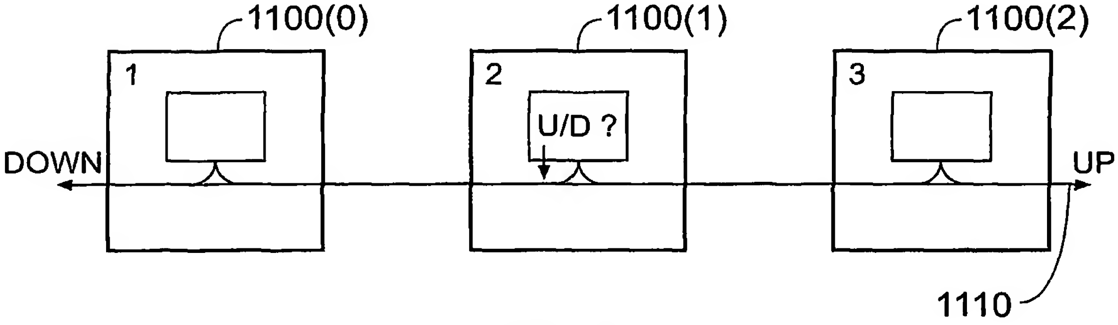

1100(0)
1100(1)
1100(2)
1
2
3
U/D ?
DOWN
UP
1110

Fig. 11

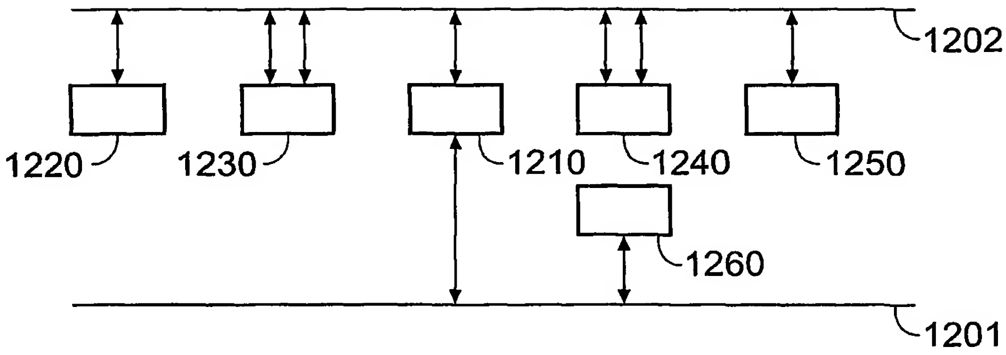

1202
1220
1230
1210
1240
1250
1260
1201

Fig. 12

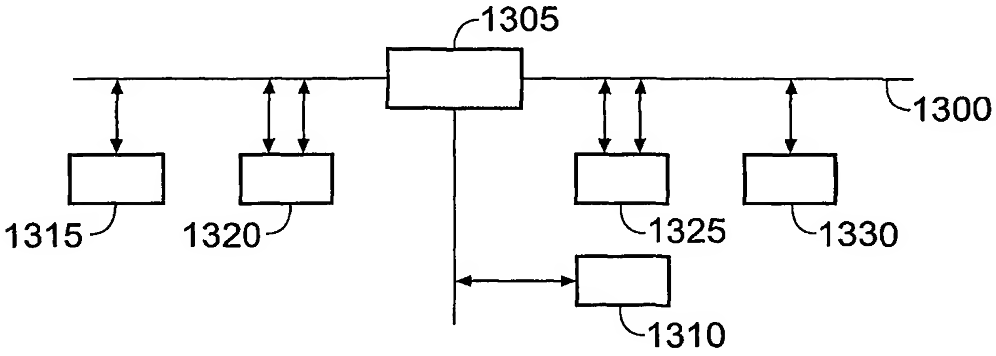

1305
1300
1315
1320
1325
1330
1310

Fig. 13

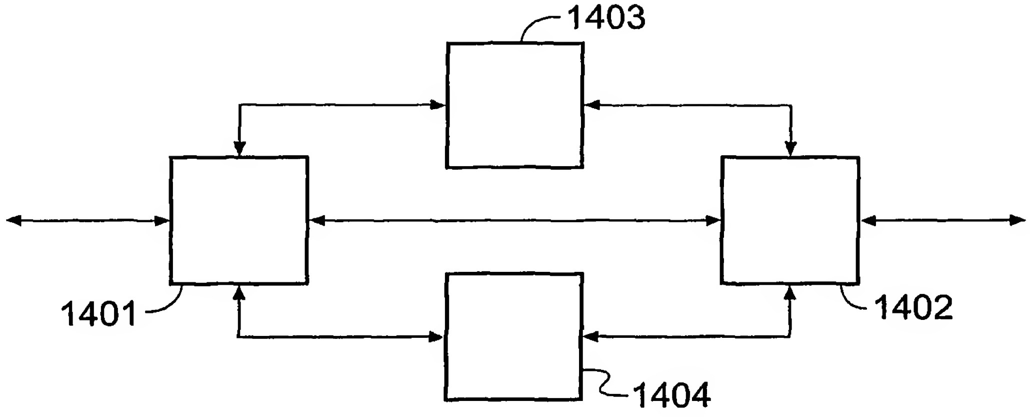

1403
1401
1402
1404

Fig. 14

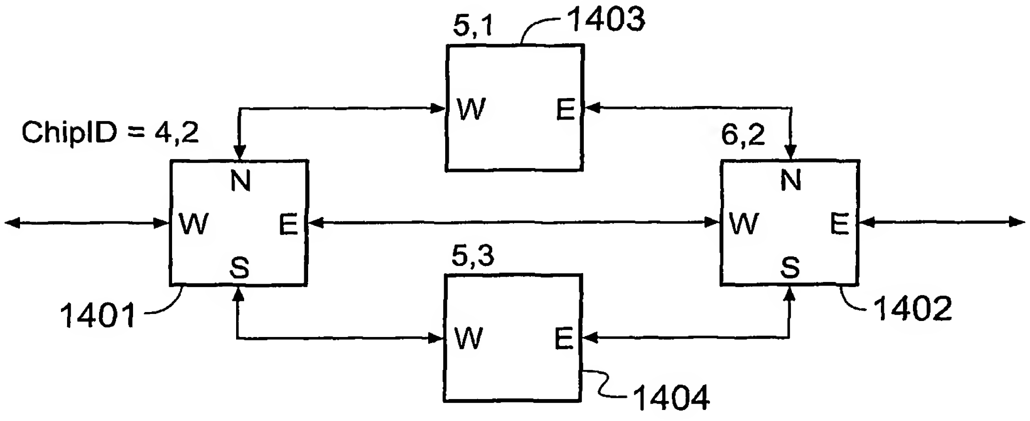

5,1
1403
W
E
ChipID = 4,2
6,2
N
W
E
S
N
W
E
S
5,3
W
E
1401
1402
1404

Fig. 15

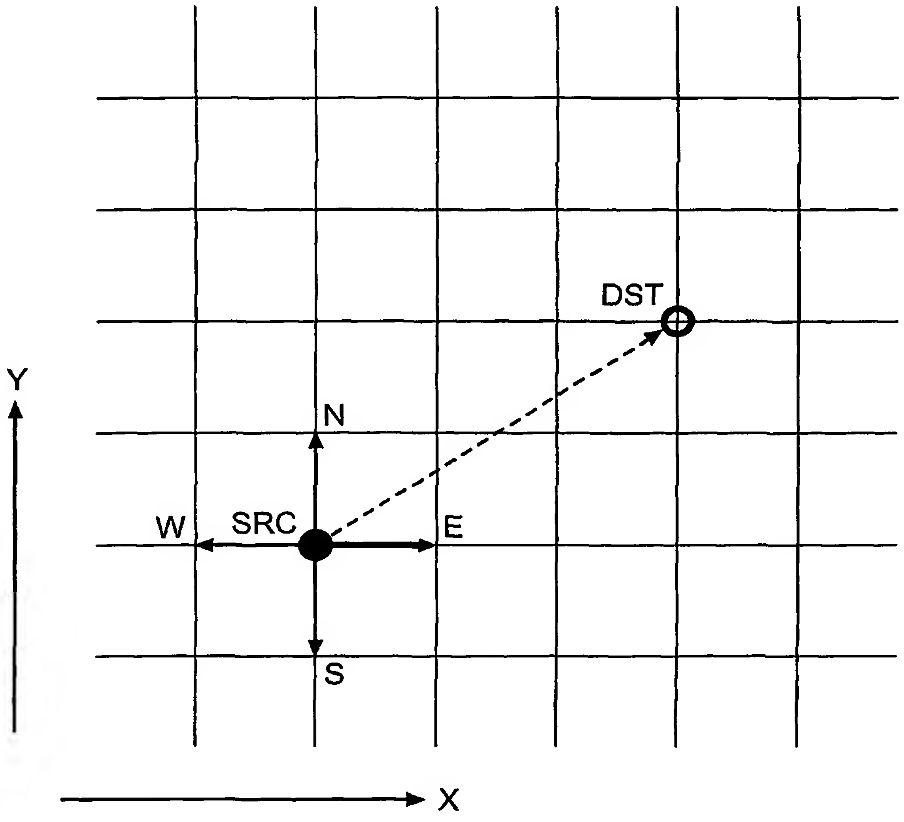


Fig. 16